**SE463 Assignment #4 Report**

Building a Program That Statically Forward Slices - Tested By Using Forward Slicing

By: Elizabeth Dixon

**Table of Contents**

* Pg. 3-4: Project Overview
* Pg. 5-6: Program Slicing + Sample Output

* Pg. 7-8: Findings Analysis & Discussion

**Project Overview**

While reviewing the material discussed in class, as well as going through the motions of this project, it was clear that Slice Testing’s promise lies in its versatility and simplicity. Its versatility, in the sense that it can be used in differing situations, relates to who uses the testing method and who can use it. Slice Testing, particularly the static, Forward Slicing method, can help software developers predict how each element of their software would act in various scenarios. In other words, Forward Slicing can help determine how their plans could be altered based on the behavior of a selected variable to slice. Using this software testing method for the first time while building my slicer.cpp program was a great learning experience. Above all else, I found myself forgetting to utilize the nuance that comes with slicing and when I consistently remembered to use it, I was reminded of how beneficial it is. You can never write too many “std::cout<< <<std::endl;” statements to check your work.

I’ll preface this by saying that writing the Forward Slicing (slicer.cpp) program was by far the hardest aspect of this project for me. No matter how productive Forward Slicing was, it could not save me from not being able to wrap my head around simple C++ logic. Nevertheless, I began by initializing the variables that I knew I could test the easiest - the simple file I/O and slice variables that would connect to the “UI” in my Terminal. I then had to consider how I would store my values as the program file or text file had to be iterated through and defined those to be sliced on. Then, I commanded to open the file, and while it was open, iterated through the input file based on the variable that would help store the whole program’s values. Then, overall, I attempted to create the logic to store the program’s values in another vector for the specified slice, testing each value at each step. Finally, the output file aspects had to be specified, making the values of the sliced variable known in a separate testingout.txt file.

**Program Slicing + Sample Output**

So, now that you have a high-level overview of the process of building this program, we can now look at the slicing and program outputs in more detail. To begin, the declarations of filenameIN and filenameOUT slice in the same manner where filenameIN and filenameOUT will affect its cin>> in the “UI” portion of the program, as well as the parameter to open in the input file. Similarly, std::ifstream infile std::ofstream outfile affects the logic for opening their respective files and closing them. Where the difference begins, however, is how infile impacts the loop that reads in the input file. It affects how while (getline(infile, sliceTemp)) is used, and assist sliceTemp in reading its respective elements and is therefore used further whenever sliceTemp is utilized (at least until the while(getline…) scope ends) and so on. sliceTemp then affects infile’s behavior as we know, it affects what values sliceCode pushes to itself and therefore the values it returns when it is referenced, it can be used to pull certain values from the whole text file (i.e. to find instances of the input slice variable), and so on. Importantly, the sliceVar is a very important input that heavily affects the logic of the program: it can be found affecting how sliceTemp works when it is called to compare file values vs. variable logic, it is used to push the variables slice’s values to a vector, and build the logic for the outfile in terms of sliced line number connection to file code, along with storedSliceVa.. The vectors infileLineNums, sliceCode, and storeSlice all function in related ways - they’re all attached to their respective variables sliceCount, sliceTemp, and sliceVar and push\_back the variable’s values based on the conditional logic for slicing. They can also be used for testing to test the values of their elements and such.

The sample output is not exactly indicative of proper static Forward Slicing, but rather the process in which vector values are stored and returned. The official testingout.txt file demonstrates the cin of sliceVar. Please see the folder “slice samples” in the of my repository: se463\_fall2024/A4.

**Findings Analysis & Discussion**

Overall, I saw the benefits of the Slicing software testing method in general, but also in the Static Forward Slicing technique. As mentioned before, the value of this technique lies in its simplicity and versatility. In terms of versatility, although it’s not as readable as a Program Graph per say, I feel as though the results of slices could easily be relayed to someone who doesn’t necessarily know what the whole of the program is doing by looking at the code. The cause and effect nature of this method is very accessible and I like that about it. It makes something that appears complicated feel more concrete and plausible - and it requires that each element of a program has tangible value. It makes me question, if a variable or entity is ineligible to be sliced, is it worth its place in a piece of software?

Although unfinished, writing a program for this method very much assisted me in knowing its true meaning conceptually. In general when writing a program, I know what I have to do but it’s a matter of knowing how to do it - and slicing can help one to understand that - even if they can’t write a program that carries out its function. For this reason, I appreciate the Slicing techniques and I also wonder what it would be like to try and code a program that uses Backwards Slicing or even Dynamic slicing, just to see how different it would be. I also wonder if it would be as plausible to use Backwards Slicing while building a program from the ground up as it would be for Forward Slicing. However, as I’m looking back at my code for this Assignment, I’m technically using Backwards Slicing as I try to figure out how to implement my conditionals correctly. Nonetheless, you always learn something new in Software Testing.